

# **Ciclo de Vida do Desenvolvimento de Software (SDLC)**

UniSenac campus Pelotas

Prof<sup>a</sup> Bruna Ribeiro

email: [brgribeiro@senacrs.com.br](mailto:brgribeiro@senacrs.com.br)

# Ciclo de Vida do Desenvolvimento de Software (SDLC)

- ❑ O ciclo de vida do sistema, ou ciclo de vida do desenvolvimento do software (SDLC) é uma estrutura que descreve as etapas necessárias para desenvolver, implementar e manter sistemas de software.
- ❑ Cada etapa possui atividades específicas que ajudam a garantir que o software atenda aos requisitos.

# Ciclo de vida



Fonte: ESPINHA, R., 2024

Disponível em:

<https://artia.com/blog/projeto-de-desenvolvimento-de-software/>

# Ciclo de Vida - Etapas

## ☐ Planejamento:

- ☐ Objetivo: Definir os objetivos do projeto e desenvolver um plano de ação. Esta fase envolve a análise das necessidades dos stakeholders, a definição do escopo do projeto, a alocação de recursos, e a criação de cronogramas.

## ☐ Atividades:

- ☐ Identificação de necessidades e problemas
- ☐ Definição dos objetivos do sistema
- ☐ Elaboração de cronogramas e planos de projeto

# Ciclo de Vida - Etapas

## ☐ Análise de Requisitos:

☐ Objetivo: Coletar e documentar todos os requisitos funcionais e não funcionais do sistema. Esta fase é crucial para garantir que o software desenvolvido atenda às necessidades dos usuários.

## ☐ Atividades:

- ☐ Reunião com stakeholders para coleta de requisitos
- ☐ Análise de requisitos para detectar inconsistências
- ☐ Documentação dos requisitos
- ☐ Revisão e validação dos requisitos com os stakeholders

# Ciclo de Vida - Etapas

## ☐ Design:

- ☐ Objetivo: Converter os requisitos em uma arquitetura detalhada do sistema. Esta fase envolve a criação de modelos que guiarão a construção do software.

## ☐ Atividades:

- ☐ Design da arquitetura do sistema
- ☐ Design de componentes e interfaces
- ☐ Criação de protótipos, se necessário
- ☐ Definição de padrões e tecnologias a serem usados

# Ciclo de Vida - Etapas

## ☐ Desenvolvimento:

☐ Objetivo: Traduzir o design em código-fonte funcional. Os desenvolvedores escrevem e integram o código de acordo com as especificações de design.

## ☐ Atividades:

- ☐ Codificação dos módulos e funcionalidades
- ☐ Integração de componentes do sistema
- ☐ Controle de versão e gestão de configurações

# Ciclo de Vida - Etapas

## ☐ Testes:

☐ Objetivo: Verificar se o sistema funciona conforme esperado. Esta fase é essencial para garantir a qualidade do software.

## ☐ Atividades:

☐ Teste de integração

☐ Teste de sistema

☐ Teste de aceitação pelos usuários

☐ Identificação e correção de bugs



# Ciclo de Vida - Etapas

## ☐ Implantação:

- ☐ Objetivo: Colocar o sistema em produção para que os usuários finais possam utilizá-lo. A implantação pode ocorrer em várias etapas, como em um lançamento piloto seguido de um lançamento completo.

## ☐ Atividades:

- ☐ Preparação do ambiente de produção
- ☐ Treinamento dos usuários finais
- ☐ Migração de dados, se necessário
- ☐ Lançamento do sistema para produção

# Ciclo de Vida - Etapas

## ☐ Manutenção:

☐ Objetivo: Garantir que o sistema continue a funcionar conforme esperado após a implantação, corrigindo erros, realizando atualizações e melhorando o sistema conforme necessário.

## ☐ Atividades:

- ☐ Correção de bugs descobertos após a implantação
- ☐ Atualizações de software e patches de segurança
- ☐ Melhorias baseadas no feedback dos usuários
- ☐ Suporte técnico e gestão de mudanças

**E quem participa de cada fase do  
ciclo de desenvolvimento do  
software???**



# Principais funções no desenvolvimento de software

- ❑ Os papéis envolvidos no desenvolvimento de software são variados e cada um deles desempenha uma função específica dentro do ciclo de vida do desenvolvimento de software.

# Principais funções no desenvolvimento de software

## ☐ Planejamento:

### ☐ Gerente de Projetos (Project Manager)

- ☐ Coordena o planejamento do projeto, define cronogramas, orçamento e recursos.

### ☐ Analista de Negócios (Business Analyst)

- ☐ Entende as necessidades do negócio e ajuda a definir os objetivos e o escopo do projeto.

### ☐ Stakeholders (Clientes, Gestores, Usuários Finais)

- ☐ Contribuem com a definição dos requisitos de alto nível e expectativas.

# Principais funções no desenvolvimento de software

## ☐ Análise de Requisitos:

### ☐ Analista de Negócios

- ☐ Coleta e documenta os requisitos funcionais e não funcionais.

### ☐ Engenheiro de Requisitos

- ☐ Detalha e especifica os requisitos do sistema.

### ☐ Product Owner (PO)

- ☐ Representa os interesses dos clientes e valida os requisitos.

### ☐ Stakeholders

- ☐ Validam e refinam os requisitos baseados nas necessidades do negócio.

# Principais funções no desenvolvimento de software

## ☐ Design:

### ☐ Arquiteto de Software

- ☐ Desenha a arquitetura do sistema, define padrões e frameworks a serem utilizados.

### ☐ Desenvolvedores

- ☐ Participam no design detalhado, ajudando a definir a estrutura técnica.

### ☐ Designer de UX/UI

- ☐ Cria protótipos e designs de interface, focando na experiência do usuário.

# Principais funções no desenvolvimento de software

## ☐ Desenvolvimento:

### ☐ Desenvolvedores (Front-end, Back-end, Full-stack)

- ☐ Escrevem e integram o código-fonte conforme o design especificado.

### ☐ Engenheiro de Software

- ☐ Desenvolve e mantém a qualidade do código e padrões de codificação.

### ☐ Especialista em DevOps

- ☐ Automatiza e gerencia o ambiente de desenvolvimento.



# Principais funções no desenvolvimento de software

## ☐ Teste:

### ☐ Engenheiro de Qualidade (QA Engineer)

- ☐ Cria planos de teste e executa testes para garantir que o software atenda aos requisitos.

### ☐ Testadores (Testers)

- ☐ Executam testes manuais e automatizados, incluindo testes funcionais e de regressão.

# Principais funções no desenvolvimento de software

## ☐ Implantação:

### ☐ Especialista em DevOps

- ☐ Gerencia a implantação do software em ambientes de produção, configura servidores e monitora a performance.

### ☐ Gerente de Projetos

- ☐ Coordena a implantação e comunicação com stakeholders.

### ☐ Suporte Técnico

- ☐ Oferece suporte durante e após a implantação, resolvendo problemas em tempo real.

### ☐ Desenvolvedores

- ☐ Prestam suporte técnico e resolvem problemas durante a implantação.

# Principais funções no desenvolvimento de software

## ☐ Manutenção:

### ☐ Desenvolvedores (Equipe de Manutenção)

- ☐ Corrigem bugs, realizam atualizações e implementam melhorias contínuas.

### ☐ Especialista em DevOps

- ☐ Monitora a saúde do sistema em produção e implementa correções de infraestrutura.

### ☐ Suporte Técnico

- ☐ Resolve problemas relatados pelos usuários e fornece assistência contínua.

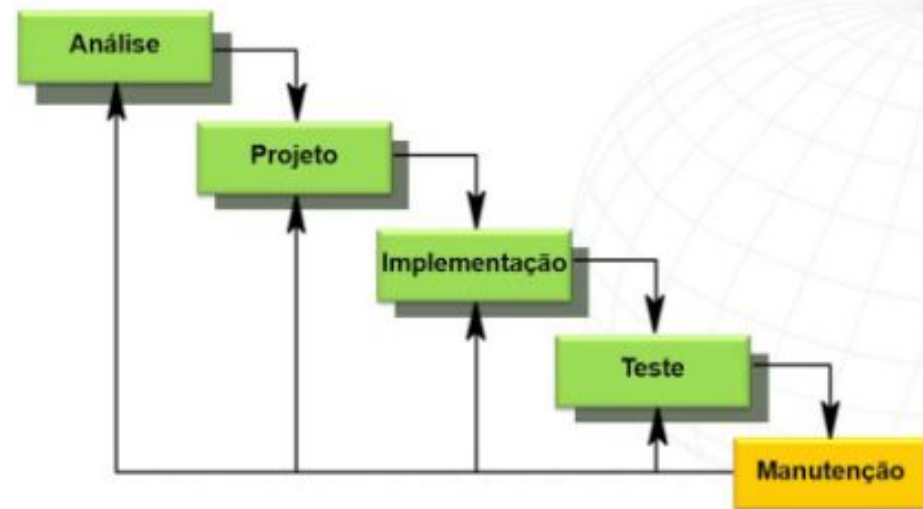
# Modelo Cascata

# Modelo Cascata

- ❑ É um dos modelos de processo de desenvolvimento de software mais antigos e tradicionais. Ele é caracterizado por uma **abordagem sequencial e linear**, onde o desenvolvimento do software ocorre em fases distintas, com cada fase sendo **concluída** antes que a **próxima comece**.

# Modelo Cascata

- ❑ Esse modelo é chamado de "cascata" devido à forma como as fases fluem de cima para baixo, como em uma cascata de água. O modelo cascata é utilizado em projetos onde os requisitos são estáveis e bem compreendidos desde o início.



# Características

- ❑ Abordagem sequencial: Cada fase ocorre em uma sequência linear, sem sobreposições.
- ❑ Fases bem definidas: O projeto é dividido em fases distintas, como levantamento de requisitos, análise, projeto, implementação, testes e manutenção.
- ❑ Saídas claras: Cada fase produz saídas documentadas, que servem como base para a fase seguinte.
- ❑ Adequado para projetos estáveis: Funciona melhor quando os requisitos do projeto estão bem definidos e mudanças são mínimas.

# Etapas do modelo cascata

- ❑ Levantamento de requisitos
  - ❑ Os requisitos do sistema são coletados e documentados, definindo as funcionalidades e características que o software deve ter.
- ❑ Análise
  - ❑ Os requisitos são analisados em detalhes para compreender como o sistema funcionará e como as partes irão interagir entre si.



# Etapas do modelo cascata

## ☐ Projeto

- ☐ Com base na análise, é criado um plano detalhado para a arquitetura do software, incluindo design de banco de dados, interfaces de usuário, estrutura de código etc.

## ☐ Implementação

- ☐ O código-fonte é escrito de acordo com o projeto e as especificações definidas anteriormente.

# Etapas do modelo cascata

## ☐ Testes

- ☐ O software é testado para verificar se ele funciona conforme o planejado e atende aos requisitos.

## ☐ Manutenção

- ☐ Após a entrega do software, a fase de manutenção ocorre para corrigir erros, fazer melhorias e atender a novos requisitos.

# Vantagens e Desafios

- **Vantagens**

- Estrutura clara e organizada.
- Adequado para projetos com requisitos estáveis.
- Fases bem definidas facilitam o planejamento.

- **Desafios**

- Pouca flexibilidade para acomodar mudanças nos requisitos.
- Atrasos em uma fase podem afetar todo o cronograma.

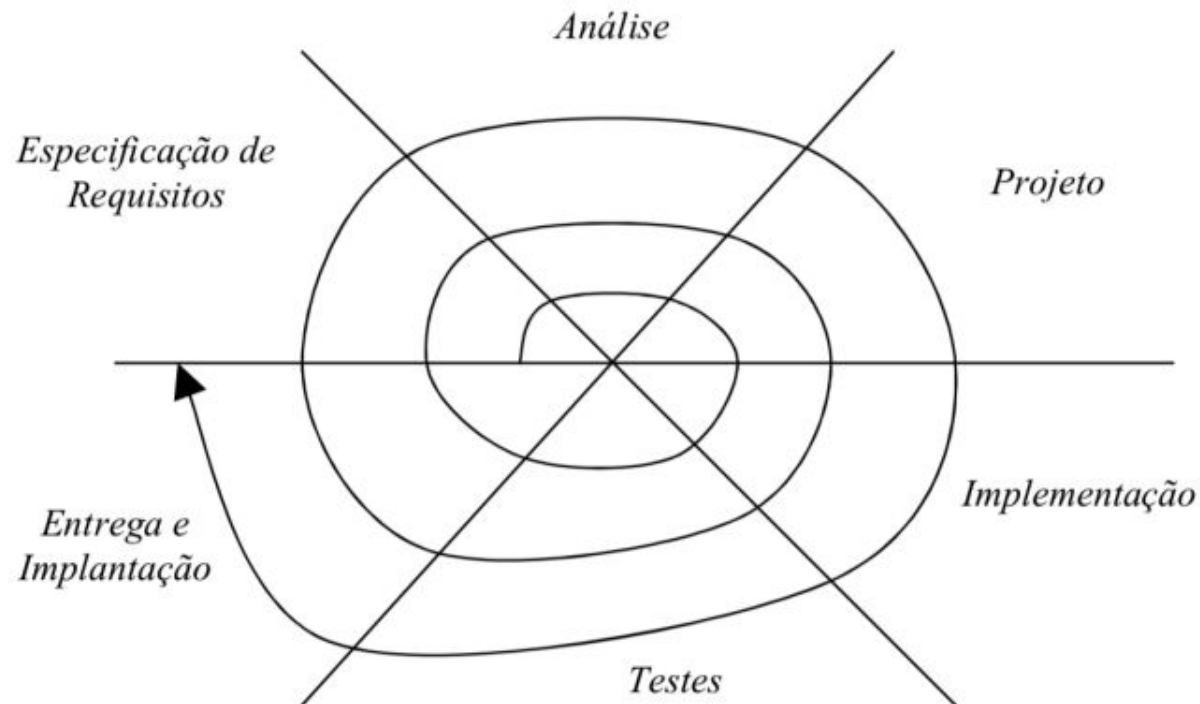
# Modelo Espiral

# Modelo Espiral

- ❑ É um modelo de processo de desenvolvimento de software que **combina** elementos de abordagens **sequenciais** e **iterativas**. Ele foi proposto por Barry Boehm em 1986 como uma maneira de lidar com a **complexidade** e os **riscos** inerentes ao desenvolvimento de software.

# Modelo Espiral

- ❑ O modelo espiral é especialmente adequado para projetos de **grande escala, complexos** ou com **alto nível de incerteza**. Os passos vão sendo repetidos até que um produto seja obtido.



# Características

- ❑ Abordagem iterativa e incremental: O desenvolvimento ocorre em ciclos, onde cada ciclo inclui atividades de planejamento, risco, engenharia e avaliação.
- ❑ Gestão de riscos: Ênfase na identificação, análise de riscos ao longo do desenvolvimento.
- ❑ Foco na avaliação contínua: Cada ciclo inclui uma avaliação detalhada das etapas anteriores, identificando problemas e ajustando a estratégia conforme necessário.
- ❑ Acomodação de mudanças: Flexibilidade para lidar com mudanças nos requisitos, com a capacidade de ajustar o projeto em cada ciclo.

# Etapas do modelo espiral

## ☐ Planejamento

- ☐ Definir objetivos, identificar riscos, alternativas e restrições, e planejar as atividades para o ciclo atual.

## ☐ Análise de Riscos

- ☐ Avaliar os riscos identificados na fase de planejamento. Decidir quais riscos serão tratados e como serão abordados.



# Etapas do modelo espiral

## ☐ Engenharia

- ☐ Executar as atividades de engenharia planejadas, como design, codificação, testes e integração.

## ☐ Avaliação

- ☐ Revisar os resultados da fase de engenharia, avaliar o progresso do projeto, identificar defeitos e problemas, e determinar se o projeto está em conformidade com os objetivos.

# Vantagens e Desafios

- **Vantagens**

- Gestão de riscos eficaz: Identificação contínua de riscos ao longo do processo.
- Adaptação a mudanças: Flexibilidade para incorporar mudanças nos requisitos.
- Abordagem progressiva: Cada ciclo resulta em um incremento funcional do software.
- Adequado para projetos complexos ou de alto risco.

- **Desafios**

- Maior complexidade gerencial: Requer avaliação constante, o que pode ser mais intensivo em termos de recursos.
- Pode ser mais demorado e custoso em comparação com modelos mais lineares.

# Prototipação

# Prototipação

- ❑ É uma abordagem essencial no processo de desenvolvimento de software, que envolve a criação de versões **iniciais** e **simplificadas** de um sistema para validar conceitos, testar **funcionalidades** e obter **feedback** dos usuários **antes** da implementação completa.

# Características

## ☐ Iterativo e Incremental:

- ☐ A prototipação geralmente ocorre em ciclos iterativos, com cada iteração refinando o protótipo com base no feedback recebido, levando a melhorias graduais.

## ☐ Interatividade:

- ☐ Um protótipo é interativo e permite que os usuários experimentem as funcionalidades-chave do sistema, muitas vezes simulando as interações reais que teriam com o produto final.

# Características

## ☐ Rapidez:

- ☐ A prototipação visa criar protótipos rapidamente para permitir a validação de ideias e conceitos em estágios iniciais do projeto.

## ☐ Foco nas Necessidades do Usuário:

- ☐ Os protótipos são projetados para atender às necessidades e expectativas dos usuários finais, ajudando a refinar os requisitos e melhorar a usabilidade.

# Tipos de Prototipação

- ❑ Prototipação de Papel (Paper Prototyping):
  - ❑ Envolve a criação de desenhos ou representações em papel do layout e fluxo de tela do sistema. É uma maneira rápida e econômica de testar ideias e interações.
- ❑ Prototipação de Baixa Fidelidade:
  - ❑ Criação de protótipos não funcionais, como wireframes, para representar visualmente a aparência geral e o fluxo do sistema.

# Tipos de Prototipação

- ❑ Prototipação de Alta Fidelidade:
  - ❑ Desenvolvimento de protótipos funcionais que simulam interações reais do usuário, muitas vezes usando ferramentas de design ou desenvolvimento.



# Vantagens e Desafios

- **Vantagens**

- Validação de Requisitos: Os protótipos ajudam a identificar requisitos incompletos, ambíguos ou mal compreendidos.
- Melhoria da Comunicação: Protótipos visuais facilitam a comunicação entre desenvolvedores, designers e stakeholders.
- Usabilidade Aprimorada: Os protótipos permitem testar a usabilidade do sistema antes da implementação.

- **Desafios**

- Possível Desvio de Escopo: A criação de protótipos pode levar a um aumento não planejado no escopo do projeto.
- Investimento de Tempo: A criação de protótipos pode consumir recursos e tempo que poderiam ser alocados para o desenvolvimento real.

# Metodologias Ágeis

# Metodologias Ágeis

- As metodologias ágeis são abordagens para o desenvolvimento de software que enfatizam a colaboração, a adaptação a mudanças, a entrega contínua de valor ao cliente e o trabalho em equipe auto-organizado.

# Metodologias Ágeis

- Elas surgiram como uma resposta aos desafios enfrentados pelos métodos tradicionais de desenvolvimento de software, como o modelo em cascata, que muitas vezes resultava em projetos atrasados, custosos e com produtos finais que não atendiam às necessidades reais do cliente.

# Principais Metodologias

## ❑ Scrum:

- ❑ O Scrum é uma das metodologias ágeis mais populares e amplamente adotadas. Ele organiza o trabalho em **sprints**, que são períodos de tempo fixos (geralmente de 2 a 4 semanas) durante os quais um conjunto específico de **funcionalidades** deve ser entregue.
- ❑ O Scrum enfatiza a **colaboração**, a transparência e a responsabilidade, com papéis definidos como Product Owner, Scrum Master e Equipe de Desenvolvimento.


# Principais Metodologias

## ❑ Extreme Programming (XP):

- ❑ O Extreme Programming é uma metodologia ágil que prioriza práticas como desenvolvimento orientado a testes (TDD), integração contínua, programação em pares e refatoração contínua do código. Ele enfatiza a **comunicação**, a simplicidade e o **feedback rápido**.

# Principais Metodologias

## Kanban:

-  Kanban é uma abordagem de gestão do trabalho baseada em princípios como visualização do trabalho, limitação do trabalho em progresso e gestão do fluxo de trabalho. Ele oferece uma abordagem flexível e adaptativa para gerenciar o trabalho em equipe, permitindo que os membros da equipe se concentrem em concluir tarefas individuais à medida que avançam pelo fluxo de trabalho.

# Benefícios

- ❑ **Flexibilidade:** As metodologias ágeis permitem uma resposta rápida a mudanças nos requisitos do projeto.
- ❑ **Entrega contínua de valor:** Priorizam a entrega contínua de software funcional e utilizável.
- ❑ **Colaboração:** Promovem a comunicação e colaboração contínuas entre os membros da equipe e com o cliente.
- ❑ **Maior satisfação do cliente:** Focam em atender às necessidades reais do cliente e adaptar-se às suas mudanças de prioridade.



# Desafios

- ❑ **Adoção Cultural:** A transição para metodologias ágeis pode exigir uma mudança cultural significativa na organização.
- ❑ **Gerenciamento de Mudanças:** Adaptar-se a mudanças frequentes nos requisitos pode ser desafiador para algumas equipes e organizações.

# Metodologias Ágeis

- ❑ As metodologias ágeis oferecem uma abordagem flexível e adaptativa que prioriza a entrega contínua de valor ao cliente e promove a colaboração e a responsabilidade dentro da equipe de desenvolvimento.
- ❑ No entanto, é importante entender que as metodologias ágeis não são uma **única solução** para todos os problemas. A chave é escolher a metodologia certa para as necessidades específicas de cada projeto e adaptá-la conforme necessário ao longo do tempo.

# Dúvidas

□ Dúvidas????



# Atividades

❑ Bora jogar???

